

The Ultimate Refactoring

Being able to transform the semantics of code written in some language and built for some specific architecture or application, i.e. a personal computer or embedded system, to some other language and/or architecture can be viewed as “the ultimate refactoring”. The perfection of this method can have huge advantages for software development, including not only how the software is developed but where that software can be deployed to. The number of opportunities for the reuse of previously written code will absolutely skyrocket. Programs will be able to be ported to other architectures and applications with the click of a button, for the most part.

Being able to transform code from one language to another will open up many opportunities for the reuse of code. Previously, software engineers writing in a particular language were confined to the reuse of code written in that same language. However, with the ability to transform code written in some other language to the one we are currently working with, the possibilities become endless as to what can be reused. Often times good programs are used for decades and are usually abandoned because of their outdated language, i.e. no one is around to remember how to code in that language. This method of refactoring can solve this problem and keep good code around.

The perfection of “the ultimate refactoring” can have a profound effect upon the portability of software to different architectures and applications. With the ability to press a button and transform code from language to language we open up the possibility to port any program to any application one's heart desires. For example, Java code can really only be run in a Java virtual machine. With this new method, if we wanted to run that same code on embedded system, we press a button, transform it into C, and everything is right with the world. This means that, for the most part, any code can be run on any architecture or application. The possibilities are endless.